Pontifícia Universidade Católica do Rio de Janeiro



GraphicsLib – Biblioteca Gráfica 2D

Documentação

Edirlei Soares de Lima

elima@inf.puc-rio.br

visionlab
visualization laboratory

Pontifícia Universidade Católica do Rio de Janeiro



Sumário

visionlab
visualization laboratory


A Ω
ALIS
GRAVE
NIL



1 Instalação e Configuração

1) Faça o download da ultima versão da biblioteca: http://www.inf.puc-rio.br/~elima/intro-prog/

2) Descompacte o arquivo GraphicsLib_v1.2.zip

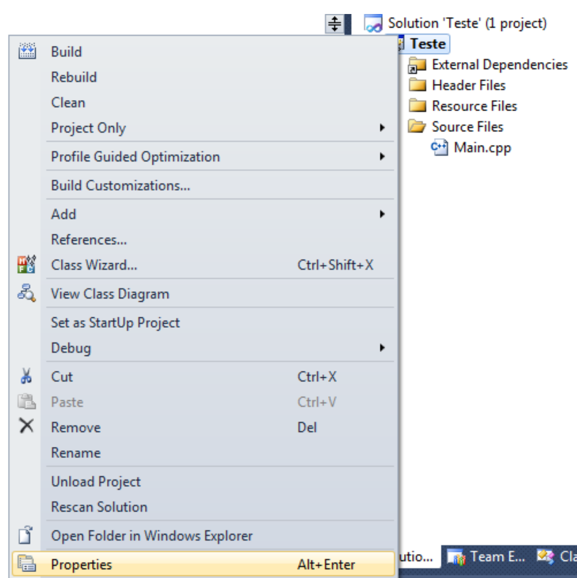
include	29/05/2012 10:41	File folder
lib	29/05/2012 10:41	File folder

3) Crie um novo projeto no **Microsoft Visual Studio 2010**. Este projeto deve ser do tipo **Win32 Console Application** na linguagem **C++.**

   Para mais informação sobre a criação de um projeto siga as instruções deste tutorial de utilização do Visual Studio 2010:

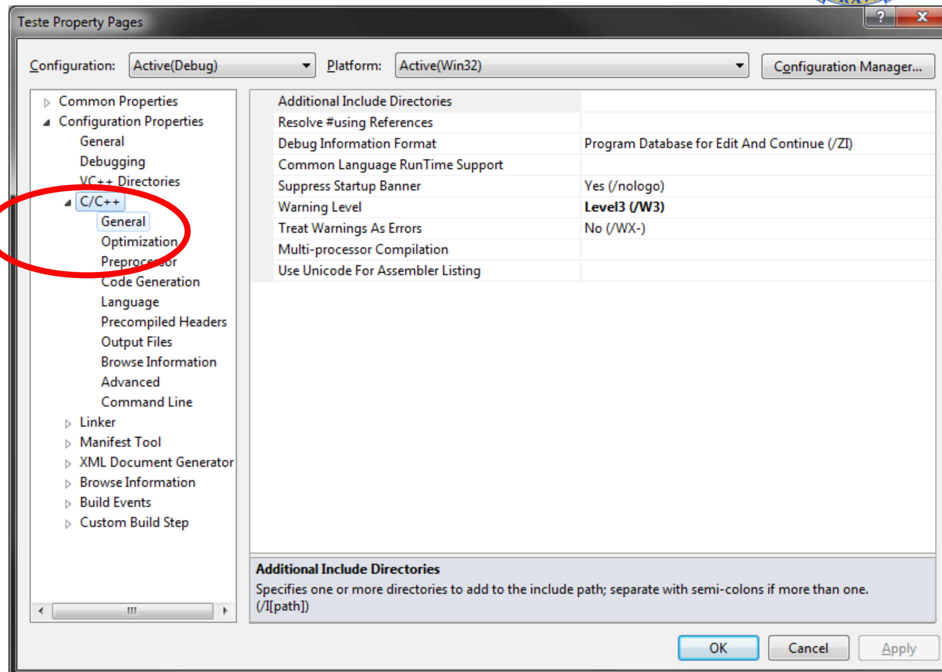
   http://www.inf.puc-rio.br/~elima/intro-prog/IntroProg_Aula_04_Introducao_Visual_Studio.pdf

4) Acesse as **propriedades do projeto** clicando com o botão da direita no nome do seu projeto.

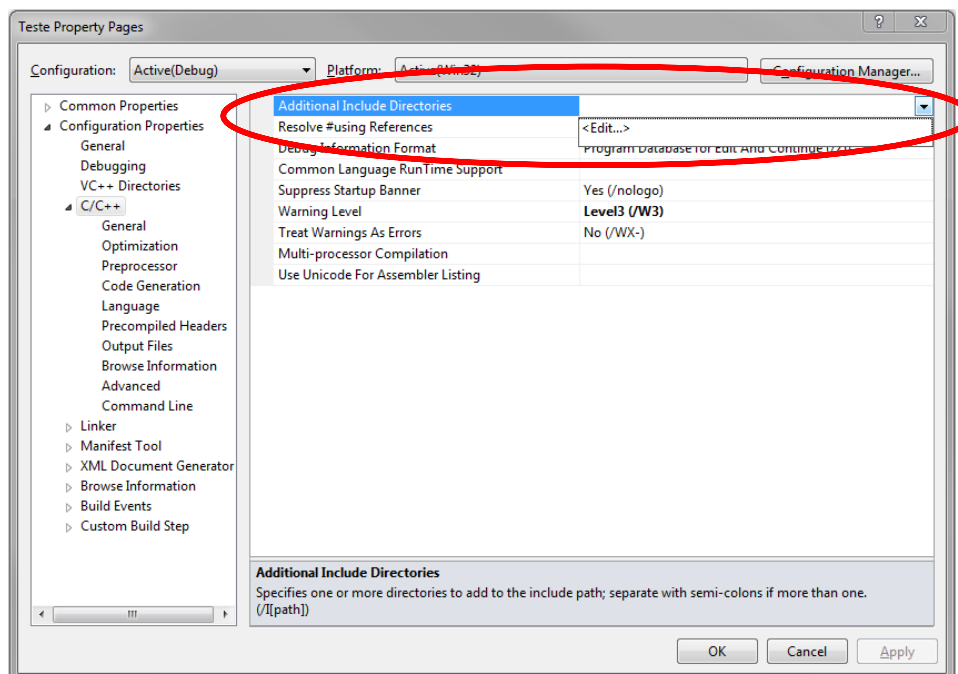


5) Selecione a opção **C/C++** e a sub-opção **General**.



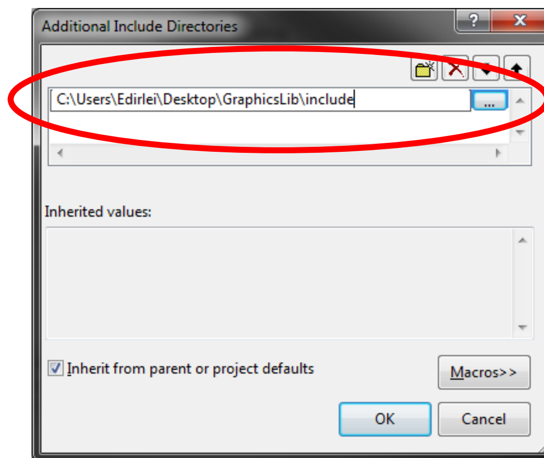


6) Selecione a opção **Additional Include Directories** e clique em **<Edit...>**



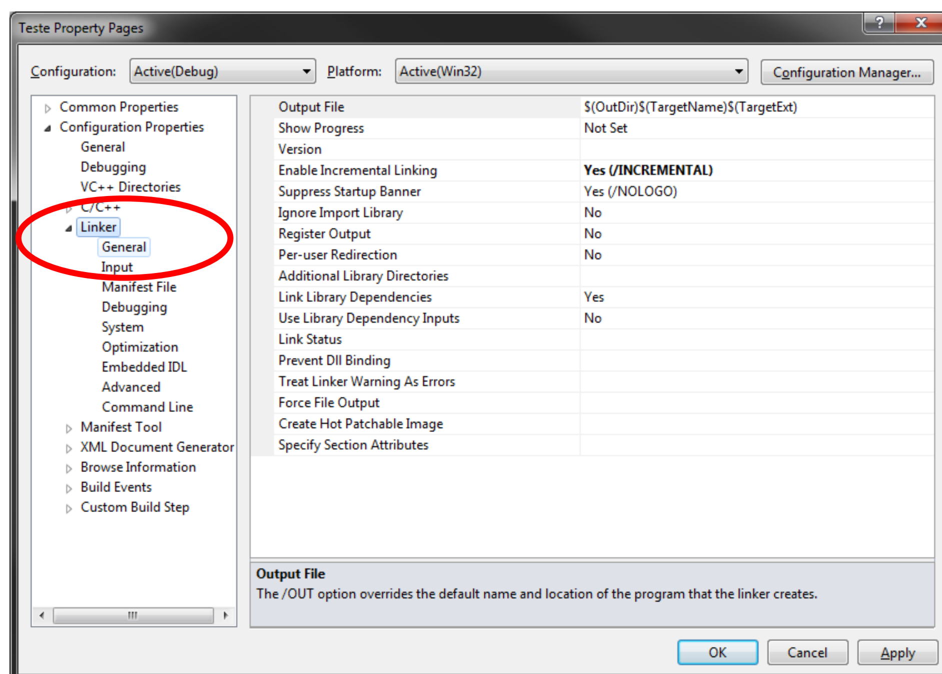


7) Selecione ou digite o caminho completo para a pasta **include** que está dento da pasta **GraphicsLib**.



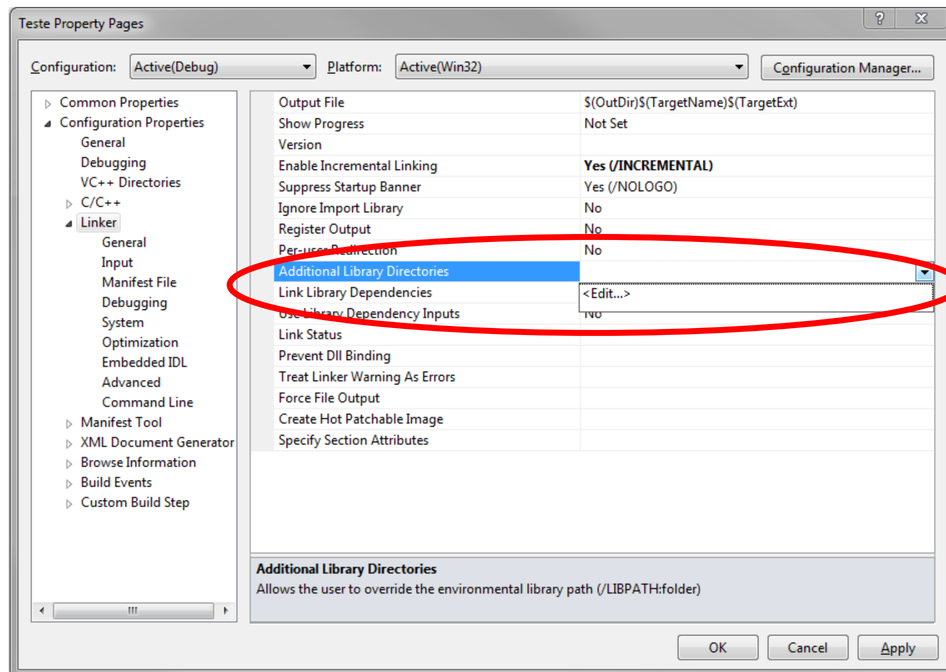
8) Clique em **OK**.

9) Selecione a opção **Linker** e a sub-opção **General**.

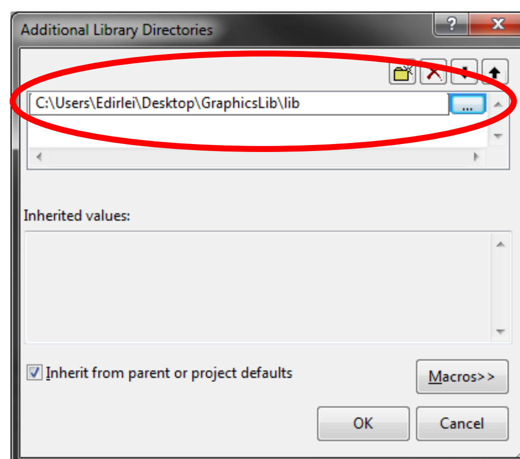




10) Selecione a opção **Additional Library Directories** e clique em **<Edit...>**



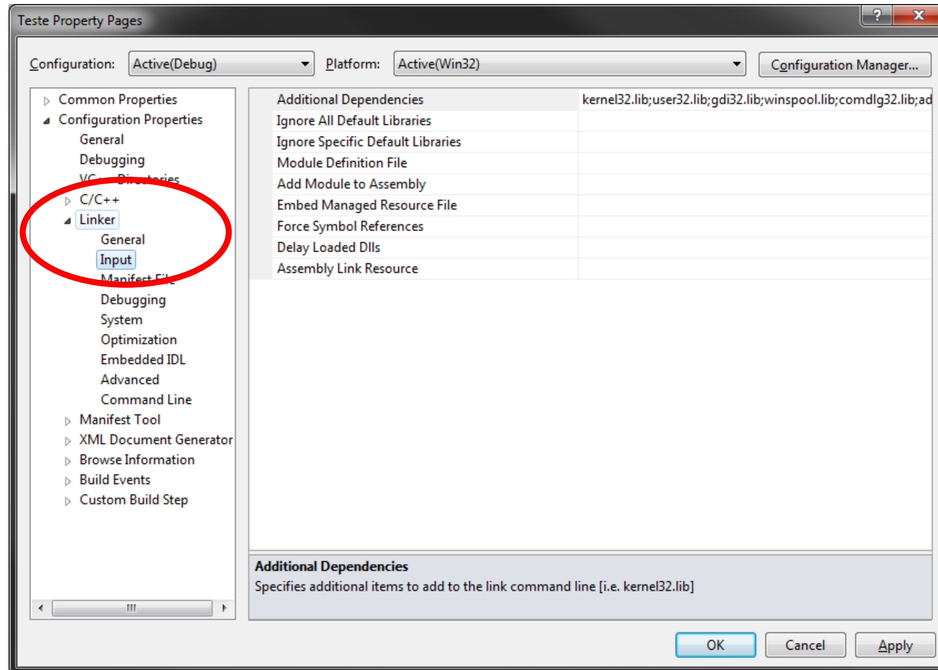
11) Selecione ou digite o caminho completo para a pasta **lib** que está dento da pasta **GraphicsLib**.



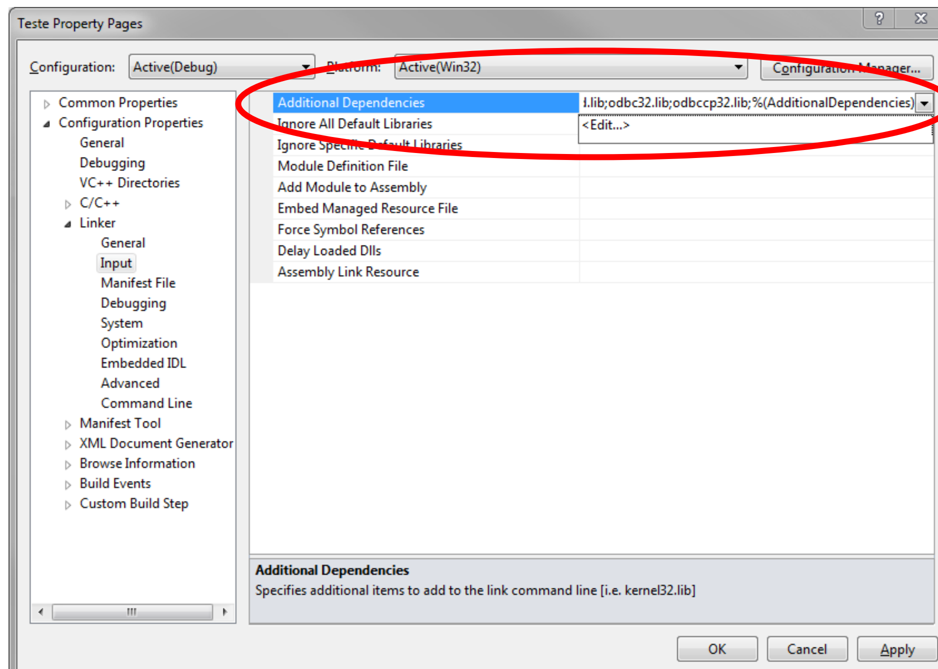
12) Clique em OK.



13) Selecione a opção **Linker** e a sub-opção **Input**.

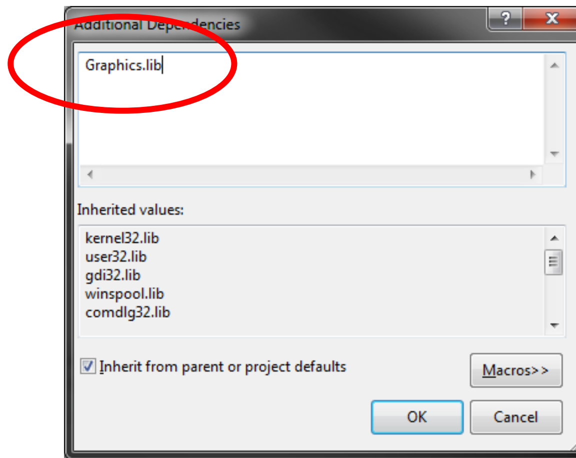


14) Selecione a opção **Additional Dependencies** e clique em **<Edit...>**



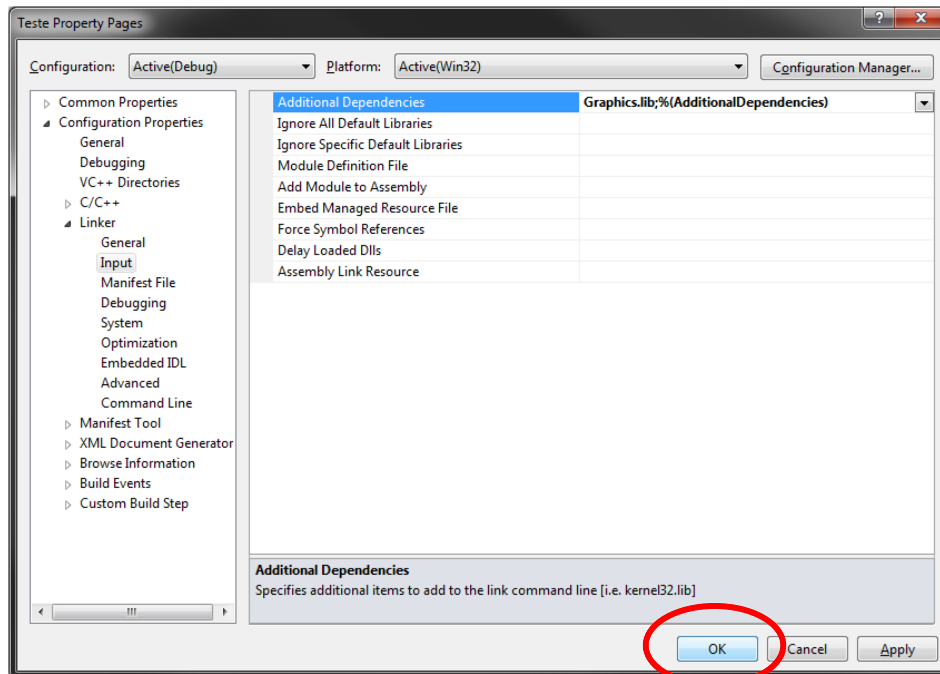


15) Digite **Graphics.lib**



16) Clique em **OK**.

17) Clique em **OK** para concluir a configuração do projeto.

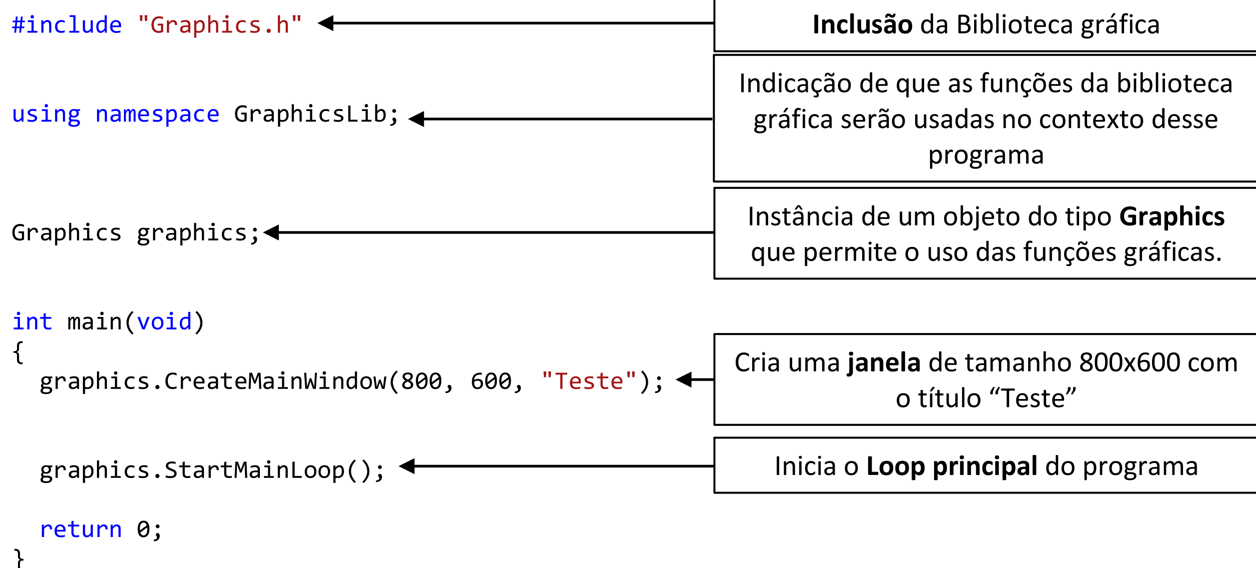




2 Manual de Utilização

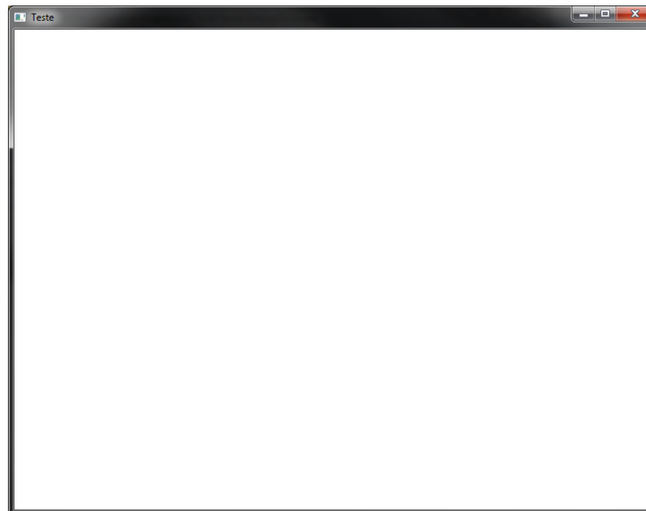
A GraphicsLib é uma biblioteca gráfica que contem um conjunto de funções para criação e manipulação de formas geométricas 2D, imagens, janelas e controle da interação pelo teclado e mouse. Com ela é possível criar jogos 2D, simulações cientificas, animações e outros aplicativos gráficos.

2.1 Estrutura de um Programa





O programa anterior simplesmente cria uma janela de tamanho 800x600 com o título “Teste” como ilustrado na figura abaixo:





2.2 Loop Principal

O **Loop Principal** consiste de uma função que é repetida enquanto o programa não for fechado pelo usuário. Todo processamento realizado pelo programa gráfico está de alguma forma ligado ao Loop Principal.

No Loop Principal deve ser programado:

- Os objetos que serão desenhados na tela e como eles serão apresentados;
- Quais animações e movimentos os objetos terão.
- Toda a lógica do programa.

Para criar o Loop Principal do programa é necessário criar uma função que será utilizada como Loop Principal. Em seguida é necessário indicar que a função criada será o Loop Principal do programa.

Exemplo:

```
#include "Graphics.h"

using namespace GraphicsLib;

Graphics graphics;

void MainLoop()
{

  graphics.SetColor(0,255,0);


  graphics.FillRectangle2D(100, 100, 400, 200);

}

int main(void)
{
  graphics.CreateMainWindow(800, 600, "Teste");

  graphics.SetMainLoop(MainLoop);

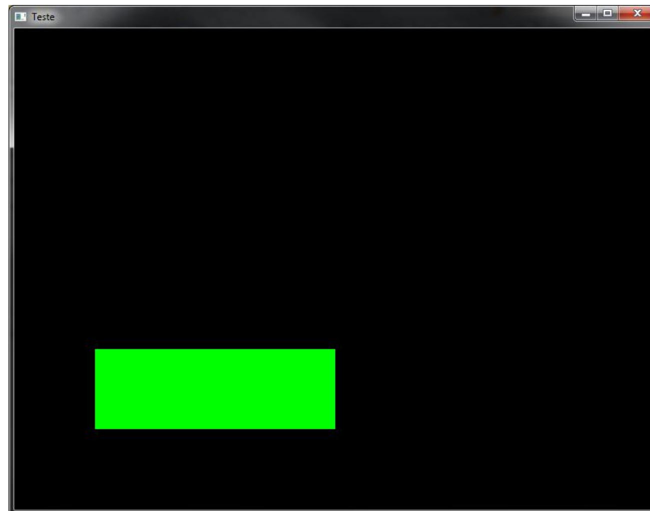
  graphics.StartMainLoop();

  return 0;
}
```

- `void MainLoop()` ← **Função** que será usada como **Loop Principal** do programa
- `graphics.SetColor(0,255,0);` ← Define a **cor** que será utilizada para desenhar objetos na tela (Formato RGB)
- `graphics.FillRectangle2D(100, 100, 400, 200);` ← Desenha um **retângulo** preenchido iniciando na posição (100,100) e indo até (200,400)
- `graphics.SetMainLoop(MainLoop);` ← Define que a função **MainLoop** será o **Loop Principal** do programa



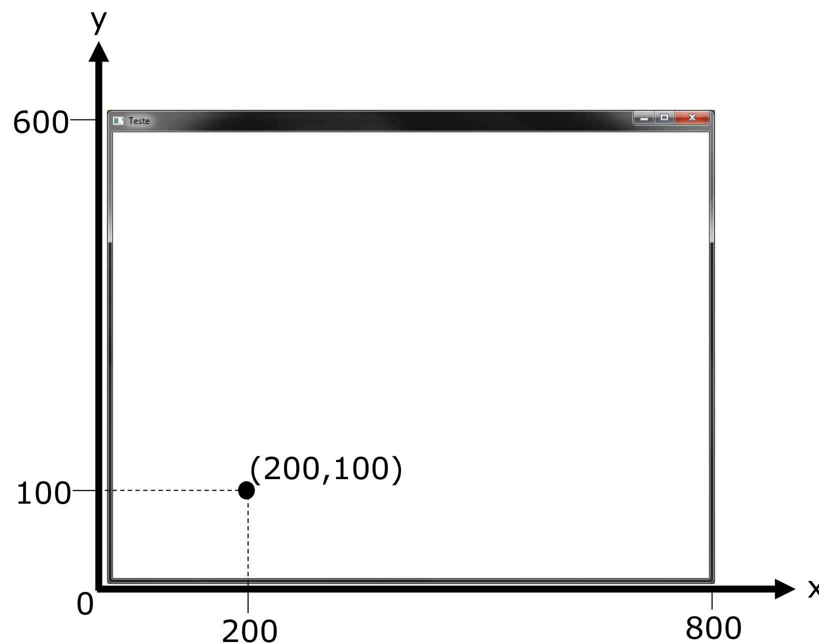
O programa anterior desenha na tela um retângulo preenchido iniciando na posição (100,100) e indo até (200,400) na cor verde como ilustrado na figura abaixo:





2.3 Coordenadas de Tela

As coordenadas de tela são definidas no sistema de coordenadas cartesiano, onde o canto inferior esquerdo da tela do programa é definido na coordenada X=0 e Y=0. Esse sistema de coordenadas é ilustrado na figura abaixo:





2.4 Desenho de Primitivas Geométricas

A GraphicsLib fornece um conjunto de funções para o desenho de primitivas geométricas básicas. As próximas seções detalham essas funções.

2.4.1 Ponto

Sintaxe:

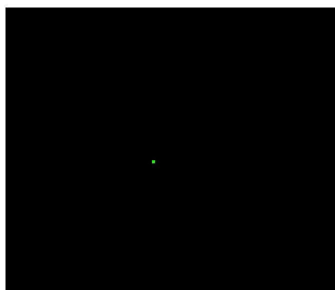
```
void DrawPoint2D(int x, int y);
```

Exemplo:

```
graphics.DrawPoint2D(200, 200);
```

Desenha um ponto na posição (200, 200) da tela.

Ilustração:





2.4.2 Linha

Sintaxe:

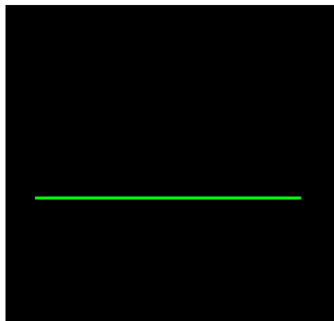
```
void DrawLine2D(int x1, int y1, int x2, int y2);
```

Exemplo:

```
graphics.DrawLine2D(100, 100, 200, 100);
```

Desenha uma linha saindo da posição (100, 100) e indo até a posição (200, 100)

Ilustração:





2.4.3 Círculo

Sintaxe:

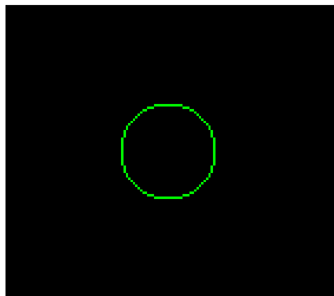
```
void DrawCircle2D(int x, int y, int radius);
```

Exemplo:

```
graphics.DrawCircle2D(200, 200, 20);
```

Desenha um círculo de raio 20 na posição (200, 200) da tela.

Ilustração:





2.4.4 Círculo Preenchido

Sintaxe:

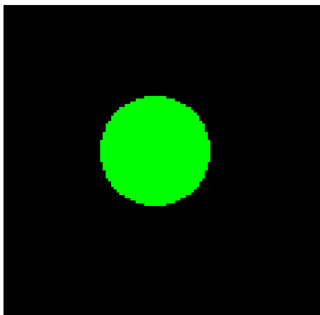
```
void FillCircle2D(int x, int y, int radius, int segments);
```

Exemplo:

```
graphics.FillCircle2D(200, 200, 20, 30);
```

Desenha um círculo preenchido de raio 20 com 30 segmentos na posição (200, 200) da tela. Quantos mais segmentos, mais redondo o circulo será.

Ilustração:





2.4.5 Retângulo

Sintaxe:

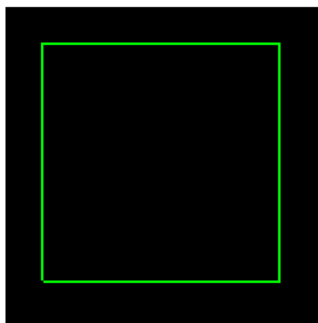
```
void DrawRectangle2D(int x1, int y1, int x2, int y2);
```

Exemplo:

```
graphics.DrawRectangle2D(100,100,200,200);
```

Desenha um retângulo iniciando na posição (100, 100) e indo até a posição (200, 200).

Ilustração:





2.4.6 Retângulo Preenchido

Sintaxe:

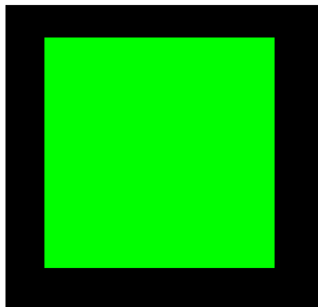
```
void FillRectangle2D(int x1, int y1, int x2, int y2);
```

Exemplo:

```
graphics.FillRectangle2D(100,100,200,200);
```

Desenha um retângulo preenchido iniciando na posição (100, 100) e indo até a posição (200, 200).

Ilustração:





2.4.7 Triângulo

Sintaxe:

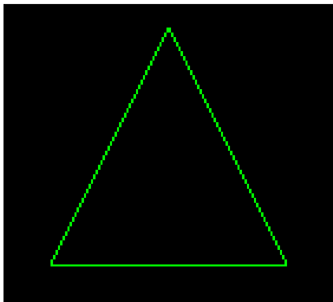
```
void DrawTriangle2D(int x1, int y1, int x2, int y2, int x3, int y3);
```

Exemplo:

```
graphics.DrawTriangle2D(100,100,200,100,150,200);
```

Desenha um triângulo com o primeiro ponto na posição (100, 100), segundo ponto na posição (200, 100) e terceiro ponto na posição (150, 200).

Ilustração:





2.4.8 Triângulo Preenchido

Sintaxe:

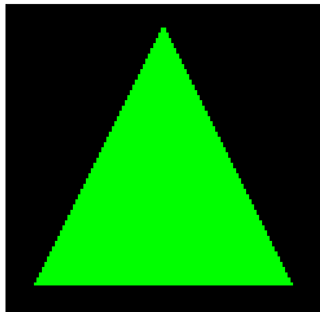
```
void FillTriangle2D(int x1, int y1, int x2, int y2, int x3, int y3);
```

Exemplo:

```
graphics.FillTriangle2D(100,100,200,100,150,200);
```

Desenha um triângulo preenchido com o primeiro ponto na posição (100, 100), segundo ponto na posição (200, 100) e terceiro ponto na posição (150, 200).

Ilustração:





2.4.9 Texto

Sintaxe:

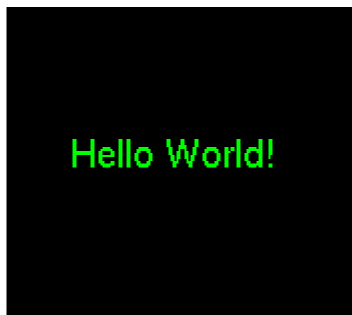
```
void DrawText2D(int x, int y, char* text);
```

Exemplo:

```
graphics.DrawText2D(100, 100, "Hello World!");
```

Escreve "Hello World!" na posição (100, 100) da tela.

Ilustração:





2.4.10 Texto (Variável Inteira)

Sintaxe:

```
void DrawTextInt2D(int x, int y, int value);
```

Exemplo:

```
graphics.DrawTextInt2D(100, 100, MinhaVariavel);
```

← Escreve o valor atual armazenado na variável inteira "MinhaVariavel" na posição (100, 100) da tela

Ilustração:





2.4.11 Texto (Variável Float)

Sintaxe:

```
void DrawTextFloat2D(int x, int y, float value);
```

Exemplo:

```
graphics.DrawTextFloat2D(100, 100, Valor3);
```

Escreve o valor atual armazenado na variável float "Valor3" na posição (100, 100) da tela.

Ilustração:





2.4.12 Modificando a Cor

Sintaxe:

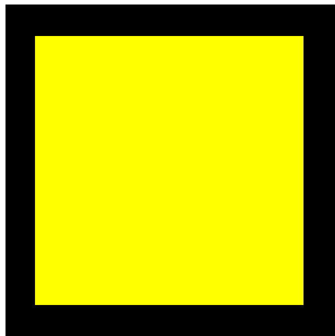
```
void SetColor(float r, float g, float b);
```

Exemplo:

```
graphics.SetColor(255, 255, 0);
```

Altera a cor que será usada para desenhar os objetos para o valor RGB (255,255,0). Ou seja, mistura o máximo de vermelho com o máximo de verde, o que resulta em amarelo.

Ilustração:





2.4.13 Modificando a Cor de Fundo da Tela

Sintaxe:

```
void SetBackgroundColor(float r, float g, float b);
```

Exemplo:

```
graphics.SetBackgroundColor(255, 255, 255);
```

Altera a cor do fundo da tela para o valor RGB (255,255,255). Ou seja, mistura o máximo de todas as cores, o que resulta em branco.

Ilustração:





2.4.14 Modificando a Largura das Linhas

Sintaxe:

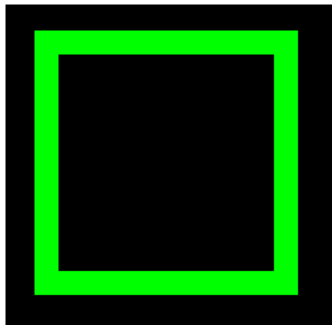
```
void SetLineWidth(float width);
```

Exemplo:

```
graphics.SetLineWidth(12);
```

Altera para 12 a largura das linhas usadas para desenhar as formas geométricas.

Ilustração:





2.5 Outras Funções

2.5.1 Criando a Janela do Programa

Sintaxe:

```
void CreateMainWindow(int sizeX, int sizeY, char title[]);
```

Exemplo:

```
graphics.CreateMainWindow(800, 600, "Nome da Janela");
```

Cria a janela principal do programa com o tamanho 800x600 e com o título “Nome da Janela”

Ilustração:





2.5.2 Executando o Programa em Tela Cheia

Sintaxe:

```
void SetFullscreen(bool enable);
```

Exemplo:

```
graphics.SetFullscreen(true);   ← Coloca o programa em tela cheia

graphics.SetFullscreen(false);  ← Remove o programa da tela cheia
```



2.5.3 Velocidade de Execução do Programa (FPS)

Sintaxe:

```
float GetFPS();
```

Exemplo:

```
fps = graphics.GetFPS();
```

Coloca o número de frames por segundo na variável fps

FPS (Frames per Second): Medida que nos indica quantos frames (imagens) consecutivos a placa de vídeo está conseguindo gerar por segundo.



2.5.4 Velocidade de Execução do Programa (ElapsedTime)

Sintaxe:

```
float GetElapsedTime();
```

Exemplo:

```
PosicaoX = PosicaoX + (Speed * graphics.GetElapsedTime());
```

> Calcula o deslocamento em X de forma independente da taxa de frames por segundo. Isso é muito importante, pois permite que a velocidade do deslocamento seja independente da velocidade que o jogo está sendo executado.



2.5.5 Largura e Altura da Janela

Sintaxe:

```
int GetScreenWidth();

int GetScreenHeight();
```

Exemplo:

```
width = graphics.GetScreenWidth();
```

Coloca a largura da tela na variável width

```
height = graphics.GetScreenHeight();
```

Coloca a altura da tela na variável height



2.6 Desenhando Imagens

Para desenhar uma imagem na tela é necessário:

1) Criar uma variável do tipo Image.

```
Image minha_imagem;
```

OBS: Sempre declare as variáveis Image como **variáveis globais**.

Exemplo:

```
#include "Graphics.h"
using namespace GraphicsLib;

Graphics graphics;
Image minha_imagem1;
Image minha_imagem2;

int main(void)
{
...
```

Variáveis Image declaradas no inicio do programa. Antes e fora da função principal ou outras funções.

2) Carregar a imagem do arquivo usando o comando LoadPNGImage.

```
minha_imagem = graphics.LoadPNGImage("Mario.png");
```

Exemplo:

```
int main(void)
{
...
  minha_imagem = graphics.LoadPNGImage("Mario.png");
...
}
```

Carrega a imagem do arquivo **Mario.png** para a variável minha_imagem.

OBS: Cada imagem deve ser carregada **apenas uma vez**. Por isso, nunca carregue a imagem diretamente de dentro do Loop Principal.



3) Desenhar efetivamente a imagem na tela usando o comando DrawImage2D.

```
graphics.DrawImage2D(200, 200, 256, 256, minha_imagem);
```

Exemplo:

```
void MainLoop()
{
...
  graphics.DrawImage2D(200, 200, 256, 256, minha_imagem);
...
}
```

Desenha a imagem "minha_imagem" na posição (200, 200) com tamanho (256, 256) na tela.



2.6.1 Carregando uma Imagem

Sintaxe:

```
Image LoadPNGImage(char *filename);
```

Exemplo:

```
Image mario;
```

Declaração da variável do tipo Image que vai armazenar a imagem

```
mario = graphics.LoadPNGImage("Mario.png");
```

Carrega o arquivo “Mario.png” para a variável “mario”

Ilustração:





2.6.2 Desenhando uma Imagem

Sintaxe:

```
void DrawImage2D(int x, int y, int width, int height, Image image);
```

Exemplo:

```
graphics.DrawImage2D(200, 200, 256, 256, mario);
```

← Desenha a imagem “mario” na posição (200, 200) com tamanho (256, 256) na tela.

Ilustração:





2.6.3 Observações importantes sobre imagens

- **Somente são aceitas imagens no formato PNG**. Mas isso não é uma limitação, o formato PNG é um dos melhores formatos para esse tipo de aplicação. A principal vantagem é que ele permite o uso de **transparência** nas imagens.

- Cerifique-se de que as imagens que serão lidas estão **dentro da pasta do seu projeto do Visual Studio**. Se preferir armazena-las em outras pastas você deve fornecer o caminho completo para o diretório onde as imagens estão para o comando LoadPNGImage.

- Se a sua imagem estiver em **outro formado** (JPG, GIF, BMP...) você deve convertê-la para o formato PNG antes de carrega-la.



2.7 Tratando Entradas do Teclado

Para poder tratar os eventos gerados pelo teclado (**teclas sendo pressionadas**) é necessário criar uma função para essa tarefa. Essa função deve ter a seguinte sintaxe:

```
void KeyboardInput(unsigned char key, int x, int y)
{
        /* Bloco de Comandos */
}
```

Também é **necessário indicar** que essa é a sua função para tratar eventos de teclado usando a função SetKeyboardInput:

```
graphics.SetKeyboardInput(KeyboardInput);
```

Dessa forma, sempre que uma tecla normal do teclado for pressionada a função **KeyboardInput** será executada e o parâmetro **key** indicará qual tecla foi pressionada. Os parâmetros **x** e **y** indicam a posição do mouse quando a tecla foi pressionada.

Exemplo:

```
void KeyboardInput(unsigned char key, int x, int y)
{
  if (key == 'f')
  {

    graphics.SetFullscreen(true);

  }
  if (key == 'w')
  {

    posicao_personagem_x = posicao_personagem_x + 2;

  }


  if (key == KEY_ESC)
  {

    exit(0);

  }
}
```

Código	Anotação
`if (key == 'f')`	Se a letra f for pressionada
`graphics.SetFullscreen(true);`	Coloca o programa em tela cheia
`if (key == 'w')`	Se a letra w for pressionada
`posicao_personagem_x = posicao_personagem_x + 2;`	Incrementa em +2 uma variável que representa a posição de um personagem
`if (key == KEY_ESC)`	Se a letra esc (código 27) for pressionada
`exit(0);`	Fecha o programa



Algumas **teclas especiais**, como por exemplo as setas direcionais do teclado, requerem o uso de outra função especifica para elas. Essa função deve ter a seguinte sintaxe:

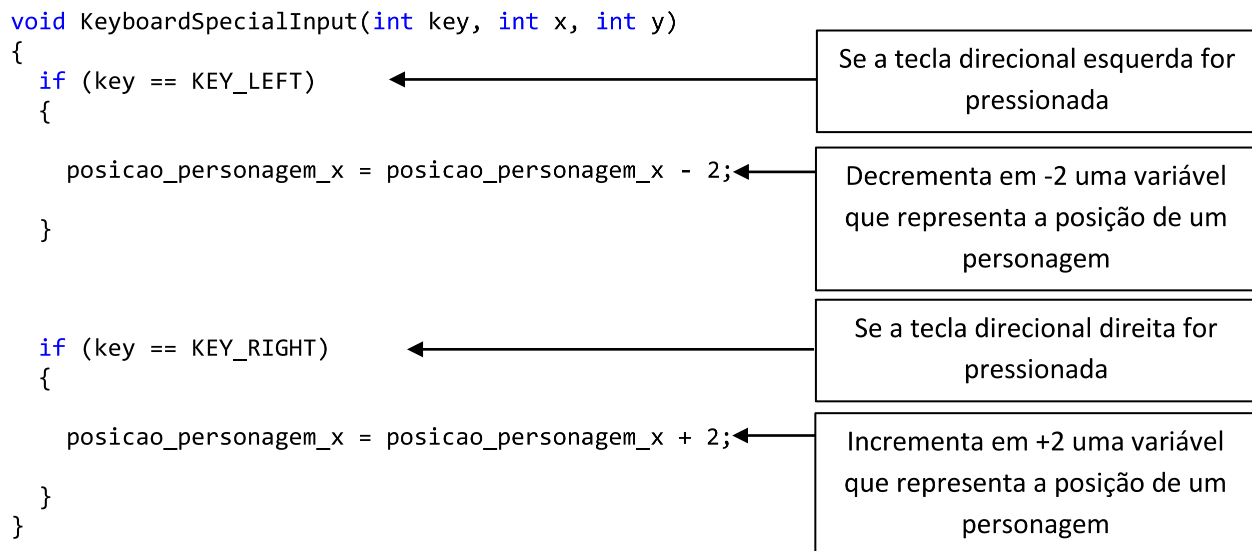
```
void KeyboardSpecialInput(int key, int x, int y)
{
        /* Bloco de Comandos */
}
```

Também é **necessário indicar** que essa é a sua função para tratar eventos de teclado especiais usando a função SetKeyboardSpecialInput:

```
graphics.SetKeyboardSpecialInput(KeyboardSpecialInput);
```

Dessa forma, sempre que uma tecla especiais do teclado for pressionada a função **KeyboardSpecialInput** será executada e o parâmetro **key** indicará qual tecla foi pressionada. Os parâmetros **x** e **y** indicam a posição do mouse quando a tecla foi pressionada.

Exemplo:




```
void KeyboardSpecialInput(int key, int x, int y)
{
  if (key == KEY_LEFT)
  {

    posicao_personagem_x = posicao_personagem_x - 2;

  }


  if (key == KEY_RIGHT)
  {

    posicao_personagem_x = posicao_personagem_x + 2;

  }
}
```



Os códigos das **teclas especiais** são os seguintes:

- KEY_LEFT
- KEY_UP
- KEY_RIGHT
- KEY_DOWN
- KEY_PAGE_UP
- KEY_PAGE_DOWN
- KEY_HOME
- KEY_END
- KEY_INSERT
- KEY_ESC
- KEY_F1
- KEY_F2
- KEY_F3
- KEY_F4
- KEY_F5
- KEY_F6
- KEY_F7
- KEY_F8
- KEY_F9
- KEY_F10
- KEY_F11
- KEY_F12



2.8 Tratando Cliques do Mouse

Para poder tratar os eventos gerados pelo mouse (**cliques do mouse**) é necessário criar uma função para essa tarefa. Essa função deve ter a seguinte sintaxe:

```
void MouseClickInput(int button, int state, int x, int y)
{
       /* Bloco de Comandos */
}
```

Também é necessário indicar que essa é a sua função para tratar eventos de clique do mouse usando a função SetMouseClickInput:

```
graphics.SetMouseClickInput(MouseClickInput);
```

Dessa forma, sempre que um botão do mouse for pressionado a função **MouseClickInput** será executada e o parâmetro **button** indicará qual botão foi pressionado. Os parâmetros **x** e **y** indicam a posição na tela em que mouse estava quando o clique foi realizado.

Exemplo:

```
void MouseClickInput(int button, int state, int x, int y)
{
  if ((button == MOUSE_LEFT_BUTTON)&&(state == MOUSE_STATE_DOWN))
  {

     destino_x = x;
     destino_y = y;

  }
}
```

Se o botão esquerdo do mouse foi pressionado

As variáveis destino_x e destino_y recebem a posição x e y do mouse no momento do clique, ou seja, onde o usuário clicou.

Os códigos dos **botões do mouse** são os seguintes:

- MOUSE_LEFT_BUTTON
- MOUSE_MIDDLE_BUTTON
- MOUSE_RIGHT_BUTTON

Os **estados** que estes botões podem assumir são os seguintes:

- MOUSE_STATE_DOWN
- MOUSE_STATE_UP



2.9 Tratando o Movimento do Mouse

Para poder tratar os eventos de movimento gerados pelo mouse é necessário criar uma função para essa tarefa. Essa função deve ter a seguinte sintaxe:

```
void MouseMotionInput(int x, int y)
{
       /* Bloco de Comandos */
}
```

Também é **necessário indicar** que essa é a sua função para tratar eventos de movimento do mouse usando a função SetMouseClickInput:

```
graphics.SetMouseMotionInput(MouseMotionInput);
```

Dessa forma, sempre que o mouse for movimentado pelo usuário a função **MouseClickInput** será executada e os parâmetros **x** e **y** indicaram a posição do mouse na tela.

Exemplo:

```
void MouseMotionInput(int x, int y)
{
  mouse_x = x;
  mouse_y = y;
}
```

As variáveis mouse_x e mouse_y recebem a posição x e y do mouse, ou seja, o local onde o usuário está com o cursor do



3 Exemplos

Essa seção apresenta alguns exemplos básicos de utilização da GraphicsLib.

3.1 Exemplo 01 – Uso de Primitivas Básicas

Este primeiro exemplo utiliza as primitivas básicas (ponto, linha, retângulo, triangulo e circulo) para desenhar uma casa simples.

```
#include "Graphics.h"
#include <stdio.h>

using namespace GraphicsLib;

Graphics graphics;

void MainLoop()
{

       graphics.SetColor(41, 156, 0);
       graphics.FillRectangle2D(0, 0, 800, 100);  //Desenha terreno

       graphics.SetColor(100, 100, 100);
       graphics.FillRectangle2D(200, 100, 400, 300);  //Desenha parede

       graphics.SetColor(255, 136, 0);
       graphics.FillTriangle2D(200, 300, 400, 300, 300, 450); //Desenha telhado

       graphics.SetColor(120, 76, 0);
       graphics.FillRectangle2D(300, 100, 370, 240);  //Desenha porta

       graphics.SetColor(9, 9, 100);
       graphics.FillRectangle2D(220, 150, 270, 220);  //Desenha janela

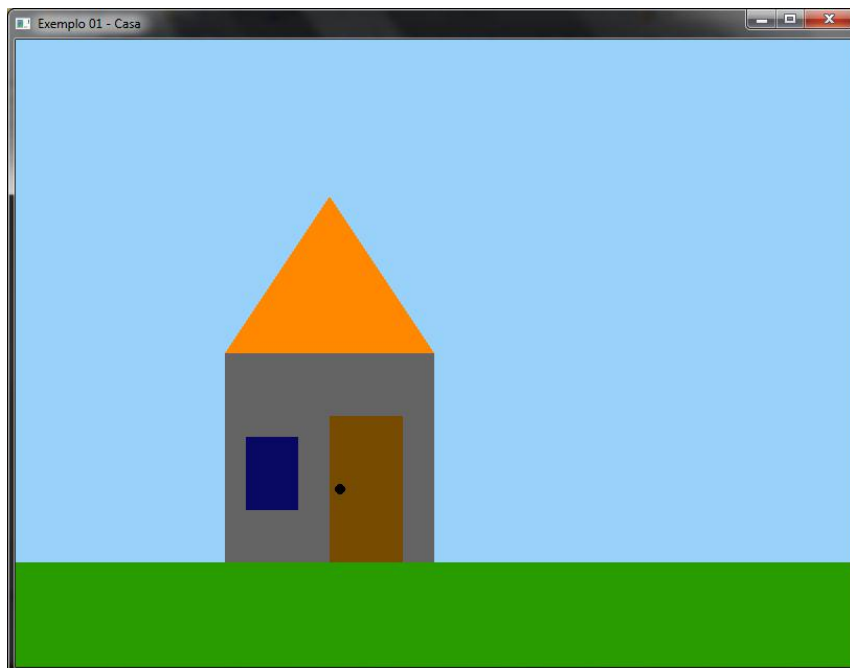
       graphics.SetColor(0, 0, 0);
       graphics.FillCircle2D(310, 170, 5, 20);  //Desenha maçaneta

}
```




```
int main(void)
{
	graphics.CreateMainWindow(800, 600, "Exemplo 01 - Casa");
	graphics.SetBackgroundColor(152,209,250);
	graphics.SetMainLoop(MainLoop);
	graphics.StartMainLoop();
	return 0;
}
```

Resultado:





3.2 Exemplo 02 – Uso de Imagens

Este exemplo utiliza imagens para criar um cenário semelhante aos cenários dos jogos da série "Super Mario". Para isso, as seguintes imagens são utilizadas:



http://www.inf.puc-rio.br/~elima/intro-prog/exemplo2_mario_imagens.zip

```
#include "Graphics.h"
#include <stdio.h>

using namespace GraphicsLib;

Graphics graphics;
Image bloco_grama;
Image bloco_cano;
Image bloco_montanha;

void MainLoop()
{
       int x;

       for (x = 0; x < 800; x+=256)
       {
              //Desenha blocos de montanhas
              graphics.DrawImage2D(x, 128, 256, 256, bloco_montanha);
       }

       //Desenha um cano
       graphics.DrawImage2D(480, 70, 128, 128, bloco_cano);

       for (x = 0; x < 800; x+=128)
       {
              //Desenha blocos de grama
              graphics.DrawImage2D(x, 0, 128, 128, bloco_grama);
       }
}
```

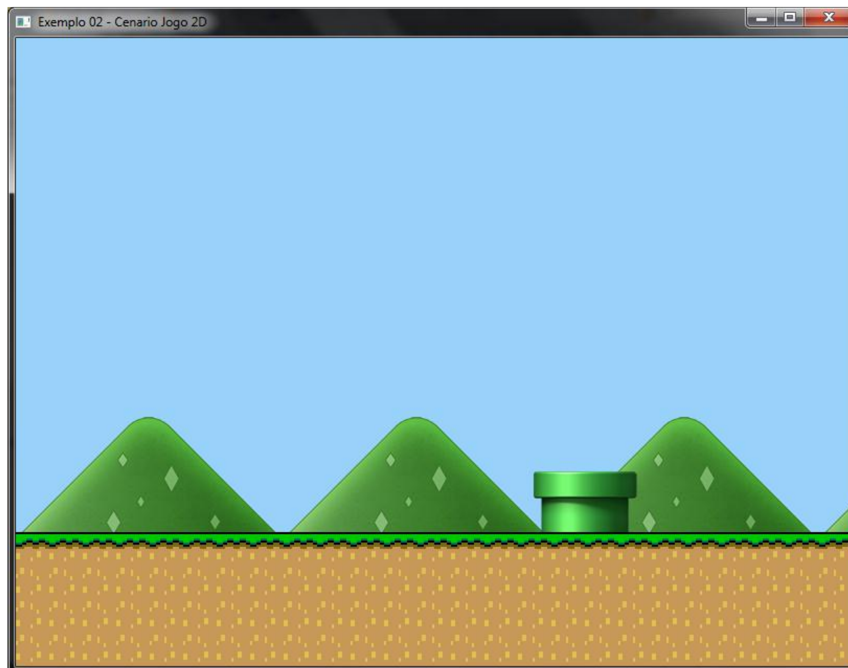



```
int main(void)
{
	graphics.CreateMainWindow(800, 600, "Exemplo 02 - Cenario Jogo 2D");
	graphics.SetBackgroundColor(152,209,250);

	//Carrega as imagens
	bloco_grama = graphics.LoadPNGImage("mario_ground.png");
	bloco_cano = graphics.LoadPNGImage("mario_pipe.png");
	bloco_montanha = graphics.LoadPNGImage("mario_background.png");

	graphics.SetMainLoop(MainLoop);
	graphics.StartMainLoop();
	return 0;
}
```

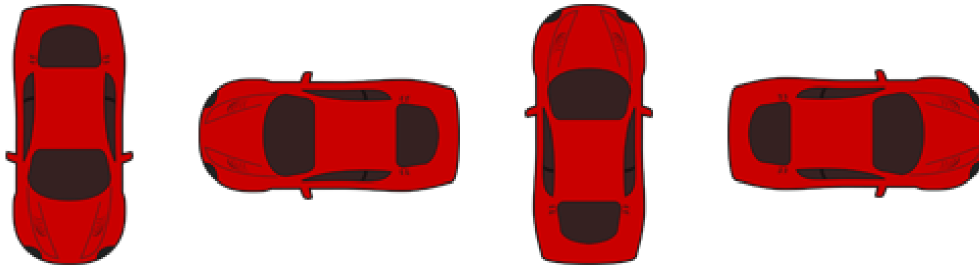
Resultado:





3.3 Exemplo 03 – Usando o Teclado

Este exemplo utiliza as funções de interação pelo teclado para criar um pequeno jogo que permite ao jogador controlar um carro usando as setas direcionais do teclado. Para isso, as seguintes imagens são utilizadas:



http://www.inf.puc-rio.br/~elima/intro-prog/exemplo3_car_imagens.zip

```
#include "Graphics.h"
#include <stdio.h>

using namespace GraphicsLib;

Graphics graphics;
Image bloco_chao;
Image carro[4];
//Direcao na qual o carro esta virando
int carro_direcao = 2;
//Posicao X do carro
int carro_x = 336;
//Posicao Y do carro
int carro_y = 236;

void MainLoop()
{
       int x, y;
       for (x = 0; x < 800; x+=256)
       {
              for (y = 0; y < 600; y+=256)
              {
                     //Desenha blocos do chao
                     graphics.DrawImage2D(x, y, 256, 256, bloco_chao);
       }
       }

       //Desenha carro
       graphics.DrawImage2D(carro_x, carro_y, 128, 128, carro[carro_direcao]);
}
```



```
void KeyboardSpecialInput(int key, int x, int y)
{
	if (key == KEY_LEFT)
	{
		carro_direcao = 1;
		carro_x = carro_x - 4;
	}
	if (key == KEY_RIGHT)
	{
		carro_direcao = 3;
		carro_x = carro_x + 4;
	}
	if (key == KEY_UP)
	{
		carro_direcao = 2;
		carro_y = carro_y + 4;
	}
	if (key == KEY_DOWN)
	{
		carro_direcao = 0;
		carro_y = carro_y - 4;
	}
}


int main(void)
{
	graphics.CreateMainWindow(800, 600, "Exemplo 03 - Controle pelo Teclado");
	graphics.SetBackgroundColor(152,209,250);

	//Carrega as imagens
	bloco_chao = graphics.LoadPNGImage("car_ground.png");
	carro[0] = graphics.LoadPNGImage("car_down.png");
	carro[1] = graphics.LoadPNGImage("car_left.png");
	carro[2] = graphics.LoadPNGImage("car_up.png");
	carro[3] = graphics.LoadPNGImage("car_right.png");

	graphics.SetKeyboardSpecialInput(KeyboardSpecialInput);
	graphics.SetMainLoop(MainLoop);
	graphics.StartMainLoop();
	return 0;
}
```



Resultado:





3.4 Exemplo 04 – Usando o Mouse

Este exemplo utiliza as funções de interação pelo mouse para criar um pequeno jogo que representar o tabuleiro de um jogo de damas. O jogador pode utilizar o mouse para colocar as peças no tabuleiro clicando no local desejado:

```
#include "Graphics.h"
#include <stdio.h>

using namespace GraphicsLib;

Graphics graphics;
int tabuleiro[8][8];

void MainLoop()
{
	int x, y;
	bool invercala = false;
	for (x = 0; x < 8; x++)
	{
		for (y = 0; y < 8; y++)
		{
			if (invercala == true)
			{
				graphics.SetColor(0,0,0);
			}
			else
			{
				graphics.SetColor(255,255,255);
			}

			graphics.FillRectangle2D(200 + (x * 50),
			                         100 + (y * 50),
			                         200 + (x * 50) + 50,
			                         100 + (y * 50) + 50);

			if (tabuleiro[x][y] == 1)
			{
				graphics.SetColor(255,0,0);
				graphics.FillCircle2D((200 + (x * 50) + 25),
				                      (100 + (y * 50) + 25),
				                       20, 20);
			}
			else if (tabuleiro[x][y] == 2)
			{
				graphics.SetColor(0,255,0);
				graphics.FillCircle2D((200 + (x * 50) + 25),
				                      (100 + (y * 50) + 25),
				                       20, 20);
```




```
                }
                invercala = !invercala;
            }
            invercala = !invercala;
        }

}

void MouseClickInput(int button, int state, int x, int y)
{
if ((button == LEFT_BUTTON)&&(state == MOUSE_STATE_DOWN))
        {
            int selecionado_x = ((x / 50) - (200 / 50));
            int selecionado_y = ((y / 50) - (100 / 50));

            if ((selecionado_x >= 0)&&
                (selecionado_x < 8)&&
                (selecionado_y >= 0)&&
                (selecionado_y < 8))
            {
                tabuleiro[selecionado_x][selecionado_y] = 1;
            }
        }
        else if ((button == RIGHT_BUTTON)&&(state == MOUSE_STATE_DOWN))
        {
            int selecionado_x = ((x / 50) - (200 / 50));
            int selecionado_y = ((y / 50) - (100 / 50));

            if ((selecionado_x >= 0)&&
                (selecionado_x < 8)&&
                (selecionado_y >= 0)&&
                (selecionado_y < 8))
            {
                tabuleiro[selecionado_x][selecionado_y] = 2;
            }
        }
}

int main(void)
{
        graphics.CreateMainWindow(800, 600, "Exemplo 04 - Controle pelo Mouse");
        graphics.SetBackgroundColor(100,100,100);

        int x, y;
        for (x = 0; x < 8; x++)
        {
            for (y = 0; y < 8; y++)
            {
                tabuleiro[x][y] = 0; //Inicializa tabuleiro
            }
        }
```



```
        graphics.SetMouseClickInput(MouseClickInput);

        graphics.SetMainLoop(MainLoop);
        graphics.StartMainLoop();
        return 0;
}
```

Resultado:

